# User's Guide for SymDyn
## Version 1.2

Karl A. Stol
Gunjit S. Bir

National Wind Technology Center
National Renewable Energy Laboratory
1617 Cole Boulevard
Golden, CO 80401-3811

Last updated on November 20, 2003

## Notice

This report was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or any agency thereof.

# Table of Contents

# 1. Introduction

SymDyn is an aeroelastic code for extracting state-space matrices for the control design of horizontal-axis wind turbines. The code runs in MATLAB v6.5 for Microsoft Windows with Simulink v2.0. Contact the authors if you have an earlier version of MATLAB, as you will require slightly different code. It is assumed that the user is familiar with the MATLAB environment to edit script code (.m files), execute commands and construct Simulink models.

SymDyn's capabilities include:

- Construction of a nonlinear aeroelastic wind turbine model.
- Calculation of periodic steady-state solutions for use as linearization points.
- Aeroelastic system linearization to form first-order state-space matrices.
- Stability analysis through Floquet theory for linear periodic systems.
- Open- and closed-loop time simulation for control performance analysis. Simulation output includes joint constraint loads and blade root loads.

The SymDyn code has been tested using only a limited collection of input cases. Certain inputs may produce erroneous results or unusual error messages. The user is advised to carefully check the input files when such an event occurs. Please direct any comments or potential bugs to karlstol@yahoo.com.

Section 2 describes the modeling assumptions in SymDyn and the aerodynamics interface. Section 3 describes the input files, analysis flow, and output variables. Section 4 provides an example study that may be used as a template for a full SymDyn analysis. This is a good place for new users to start before modifying the code. The example refers to input data (listed in Appendix B) for a two-bladed, upwind turbine. This is a model of an actual 600 kW machine, known as the Controls Advanced Research Turbine (CART), currently being studied at the National Wind Technology Center in Colorado, USA.

# 2. Model Description

## 2.1 Structural Dynamics

SymDyn uses equivalent hinge modeling assumptions for the representation of flexible turbine components (tower, drivetrain, and blades). A total of 8+$N_b$ degrees-of-freedom (DOFs) are available, where $N_b$ is the number of blades. The DOFs are all relative angular displacements measured between adjacent rigid components. See Figure 1 for a descriptive list and illustration.

An explanation of the shaft DOFs deserves special attention. The generator azimuth position, $\psi$, measures the angular position of the generator end of the drivetrain relative to the nacelle. If a gearbox is present, with gear-ratio greater than 1.0, then this angular displacement is in the low-speed frame. The actual angular displacement of the high-speed shaft (HSS) is then $\psi$×(gear-ratio). The shaft torsional deflection, $\varepsilon$, measures the angle between the rotor end of the shaft relative to the generator end. The actual angular displacement of the rotor end of the low-speed shaft (LSS) relative to the nacelle is then $\psi+\varepsilon$. Whenever 'generator speed' is mentioned, this is equivalent to $\dot{\psi}$, not necessarily the true speed of the generator, which is $\dot{\psi}$×(gear-ratio).

The user does not have to define SymDyn parameters explicitly. A preprocessor is available to generate these parameters from a general-purpose properties file. This procedure will be described in section 3.

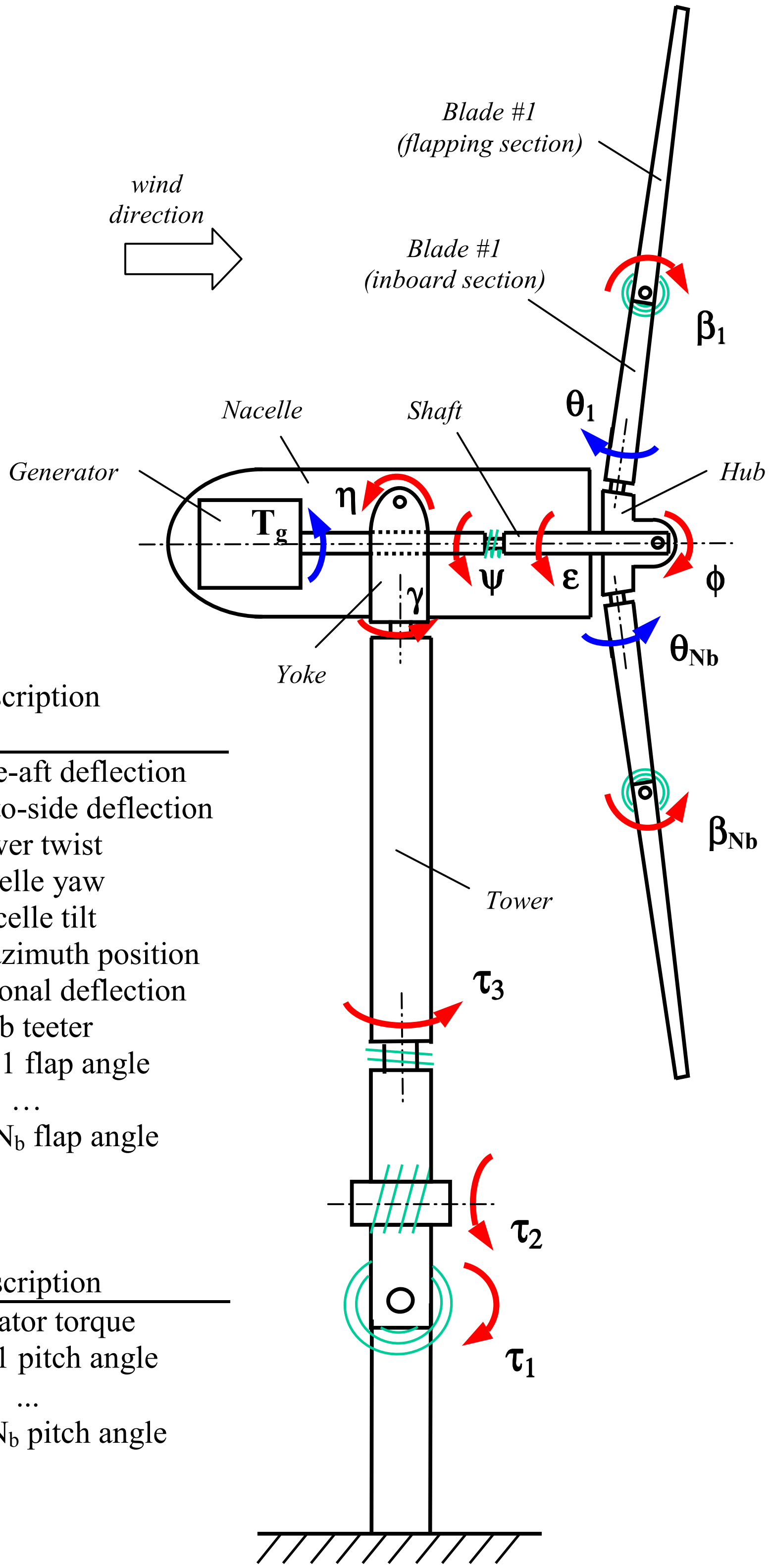
## 2.2 Aerodynamics

Aerodynamic loads are calculated through AeroDyn (currently version 12.52) Fortran subroutines, which have been compiled and linked with MATLAB interface code. AeroDyn uses blade-element-momentum (BEM) theory with models for dynamic stall, induced inflow, and tip/hub losses. The AeroDyn User's Guide [1] details the various aerodynamic options available.

Wind field data can be passed to AeroDyn in two ways during a simulation. The `usewindfile_flag` variable in the *inputsim.m* input file controls which method will be used.

1. The conventional method is to prepare input files that AeroDyn can read. These files can be categorized into hub-height data (ASCII files) or full-field turbulence data (binary files). The AeroDyn User's Guide describes the preparation of these files in detail. To enable the conventional wind file method, set `usewindfile_flag = 1` in the *inputsim.m* file. AeroDyn ignores all wind data defined in (and passed from) MATLAB.
2. The second approach is to allow MATLAB to pass wind data directly to the AeroDyn subroutines at each simulation time step. Only hub-referenced data is possible with this option. AeroDyn still requires a hub-height wind file to be defined and specified in the *aerodyn.ipt* file – the data is simply ignored. To enable this method, set `usewindfile_flag = 0` in the *inputsim.m* file and ensure the `wdata` signal in the simulation block diagram is correct.

Output from AeroDyn appears in up to four files. Three of these files (*symdyn.opt*, *error.log*, *element.plt*) are standard output that is documented in the AeroDyn User's Guide. The fourth file, *screendump.log*, contains all the text that AeroDyn would ordinarily print to the screen.



**Degrees-of-freedom:**

| DOF ref. number | Symbol | Description |
|---|---|---|
| 1 | $\tau_1$ | Tower fore-aft deflection |
| 2 | $\tau_2$ | Tower side-to-side deflection |
| 3 | $\tau_3$ | Tower twist |
| 4 | $\gamma$ | Nacelle yaw |
| 5 | $\eta$ | Nacelle tilt |
| 6 | $\psi$ | Generator azimuth position |
| 7 | $\varepsilon$ | Shaft torsional deflection |
| 8 | $\varphi$ | Hub teeter |
| 9 | $\beta_1$ | Blade #1 flap angle |
| … | … | … |
| $8+N_b$ | $\beta_{Nb}$ | Blade #$N_b$ flap angle |

**Control inputs:**

| Symbol | Description |
|---|---|
| $T_g$ | Generator torque |
| $\theta_1$ | Blade #1 pitch angle |
| … | … |
| $\theta_{Nb}$ | Blade #$N_b$ pitch angle |

**Figure 1: Illustration of SymDyn DOFs and control inputs**

Only two AeroDyn messages are passed to the MATLAB command window, to notify the user that an error or fault has occurred in the AeroDyn subroutines. The messages are

```
_ADmsg_: ERROR, Program continues...      and
_ADmsg_: Program will abort due to error.
```

If either of these messages appear, the user should consult *screendump.log* and *error.log* for the cause.

## 2.3 Verification Studies

Limited studies have been performed to verify the correct implementation of the SymDyn modeling assumptions. These studies compared time response data of a SymDyn model and an identical ADAMS® model. The only differences between the two models are the formulation of equations of motion and the time integration methods.

The most complex model that has been verified is described below.

Turbine: two-bladed machine (the example properties used in this guide)
DOFs: 8 $\{\tau_1, \tau_2, \tau_3, \psi, \varepsilon, \varphi, \beta_1, \beta_2\}$
AeroDyn options: Beddoes-Leishman dynamic stall, generalized dynamic wake, axial and tangential induction factors, Prandtl tip and hub loss.
Wind input: linear ramp in horizontal hub-height wind speed from 16 m/s to 20 m/s in 10 seconds, constant vertical wind shear exponent of 0.2.

Figure 2 presents two of the response plots for this case, which show perfect agreement.

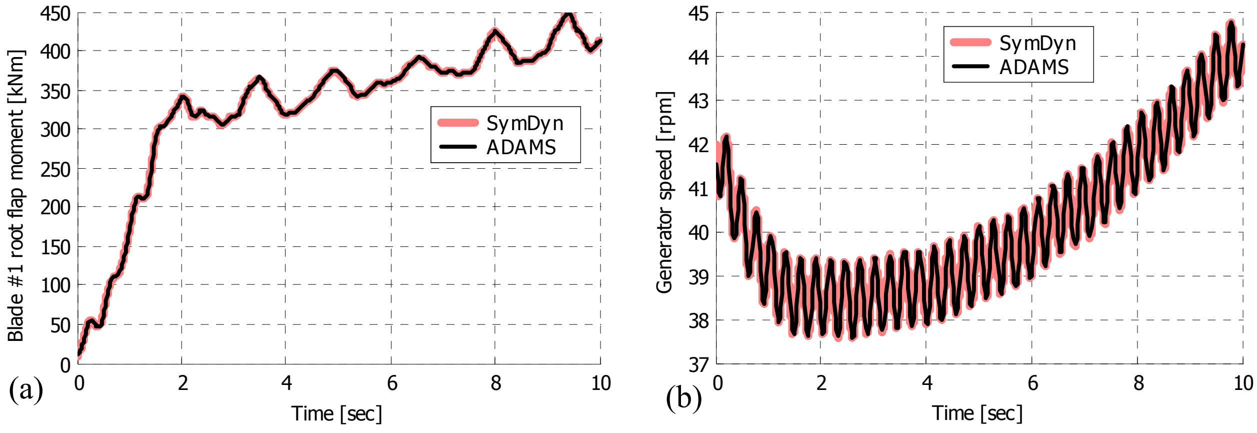


**Figure 2: Verification results (using AeroDyn v12.52)**

# 3. Running SymDyn

A flow chart for a typical SymDyn design is shown in Figure 3. The two shaded boxes at the top of the figure represent the two input files to SymDyn. The other shaded boxes represent intermediate save files. The white boxes represent script code (.m) or simulink models (.mdl) that are executed by the user in MATLAB.

## 3.1 SymDyn Preprocessor (*SymDynPP*)

The SymDyn preprocessor reads generic wind turbine properties from a file (e.g. *inputprops.m*) and generates SymDyn parameters (*SDprops.mat*) plus partial AeroDyn input data (*ADdata.ipt*). Internally, the preprocessor builds finite element models of the tower and blade components, then calculates equivalent torsional hinge locations and spring constants, consistent with SymDyn's modeling assumptions. Appendix A and reference [2] describes these assumptions in more detail.

At the MATLAB command prompt, the preprocessor is executed by entering

```
>> SymDynPP
```

Optional numerical arguments may also be given as follows:

```
>> SymDynPP(debugflag, ntelems, nbelems)
```

| | |
|---|---|
| `debugflag` | Flag specifying whether additional information is written to the command window for debugging purposes (0 = false, 1 = true, default: 0) |
| `ntelems` | Number of finite elements for the tower preprocessor model (default: 40) |
| `nbelems` | Number of finite elements for the blade preprocessor model (default: 40) |

The SymDynPP function can be called with less than three input arguments, in which case the default values are used. An empty array '[]' should be used as a place-keeper if additional arguments follow. E.g. the command

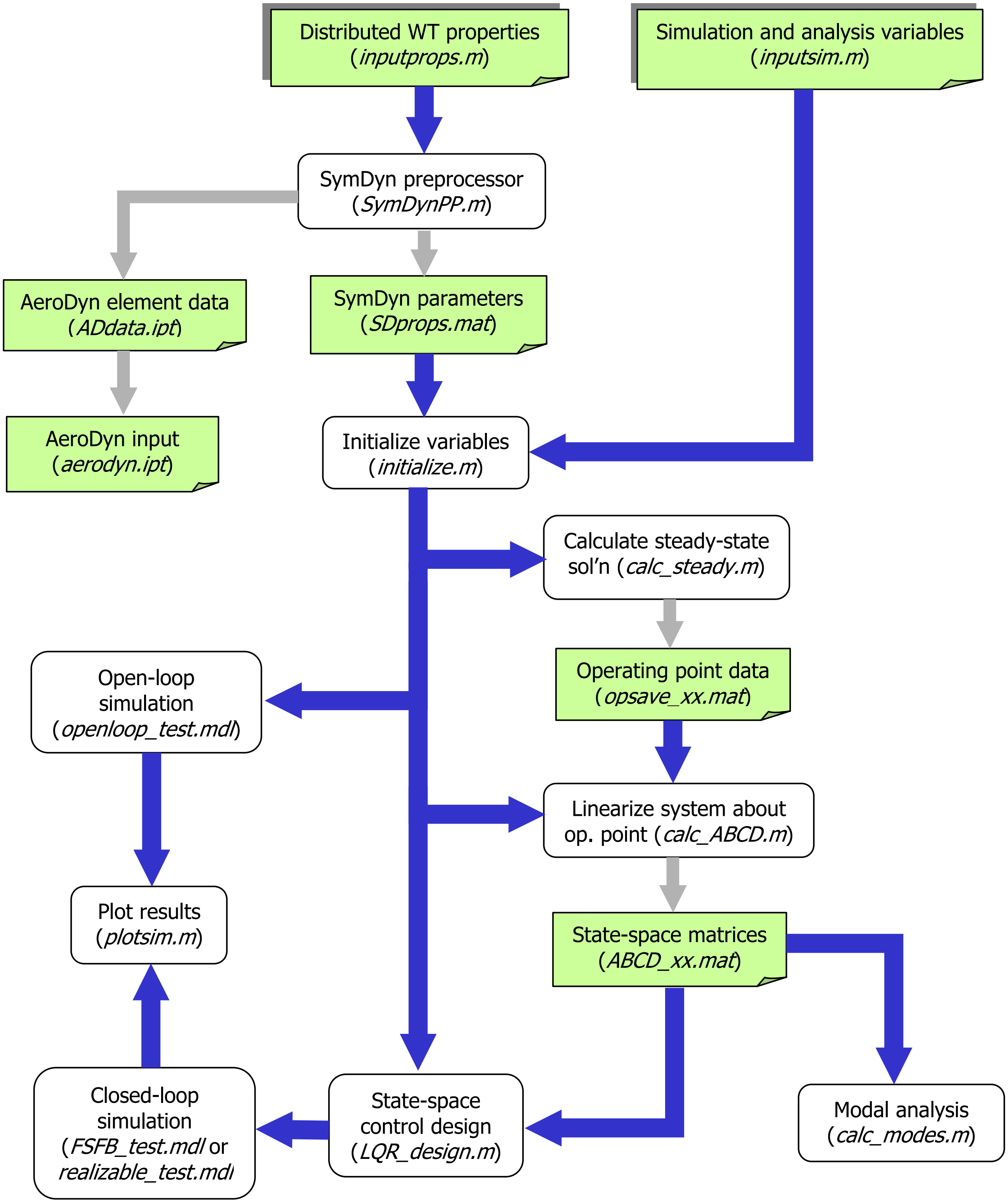
```
>> SymDynPP([], 50)
```

runs the preprocessor with the debug flag off (default), 50 tower finite elements, and 40 blade finite elements (default).

Once SymDynPP is executed, the user will be prompted for the name of the MATLAB script file that contains the turbine properties. By default this is *inputprops.m*. The user could choose a more descriptive file name (e.g. MyTurbine600kW.m) as long as it has a '.m' file extension. SymDynPP only needs to be executed before an analysis if the input file has been modified.

Upon completion of SymDynPP, two figures will appear. The first illustrates the lengths of AeroDyn elements as determined by the `aero_elem_loc` input variable (see section 3.2). The second figure plots the SymDyn model to scale, with major joint locations labeled, allowing the user to visually check that the component proportions are correct. Use 'Camera Toolbar' from the figure's 'View' pull-down menu to allow zooming and rotation of the diagram. One of the files that is written by the preprocessor, *ADdata.ipt*, contains AeroDyn input data. An example is given in

Listing 2. Lines from this file should be used to generate the *aerodyn.ipt* file required by AeroDyn, thus ensuring consistency of properties.



Distributed WT properties (*inputprops.m*)
Simulation and analysis variables (*inputsim.m*)
SymDyn preprocessor (*SymDynPP.m*)
AeroDyn element data (*ADdata.ipt*)
SymDyn parameters (*SDprops.mat*)
AeroDyn input (*aerodyn.ipt*)
Initialize variables (*initialize.m*)
Calculate steady-state sol'n (*calc_steady.m*)
Open-loop simulation (*openloop_test.mdl*)
Operating point data (*opsave_xx.mat*)
Linearize system about op. point (*calc_ABCD.m*)
Plot results (*plotsim.m*)
State-space matrices (*ABCD_xx.mat*)
Closed-loop simulation (*FSFB_test.mdl* or *realizable_test.mdl*
State-space control design (*LQR_design.m*)
Modal analysis (*calc_modes.m*)

**Figure 3: Sample SymDyn design flow**

## 3.2 Properties Input File (*inputprops.m*)

A sample *inputprops.m* file is given in Listing 1. The comment lines provide information about each property that is expected in this file. Figure 4 illustrates the geometric properties required. Single headed arrows are used as dimension lines in the figure to clarify sign conventions.

Many of the input properties use the wind direction to define a positive direction. For example, `dteeter` is defined as the distance from the tilt axis to the teeter axis, parallel to the shaft, positive downwind. Therefore, an upwind turbine (as in Listing 1) should have a negative value of `dteeter`.

The formats for tower and blade distributed data are described next.

### Tower properties

The tower distributed properties are defined by a table of data. Each row represents the properties at a given height station. The columns are:

| Symbol | Description | Units |
|---|---|---|
| $\bar{x}$ | Normalized station height. 0.0 is ground level and 1.0 is tower top. | - |
| m/L | Mass per-unit-length. | [kg/m] |
| I/L | Mass M.O.I. per-unit-length about a lateral axis. | [kg.m] |
| GJ | Section torsional rigidity. | [N·m$^2$] |
| EI | Section bending rigidity about a lateral axis. | [N·m$^2$] |

At least two rows must be defined, for $\bar{x}$ = 0.0 and $\bar{x}$ = 1.0. Linear interpolation is used to find properties at intermediate stations. An extra property is used to model the mass of tower-attached yaw drive components:

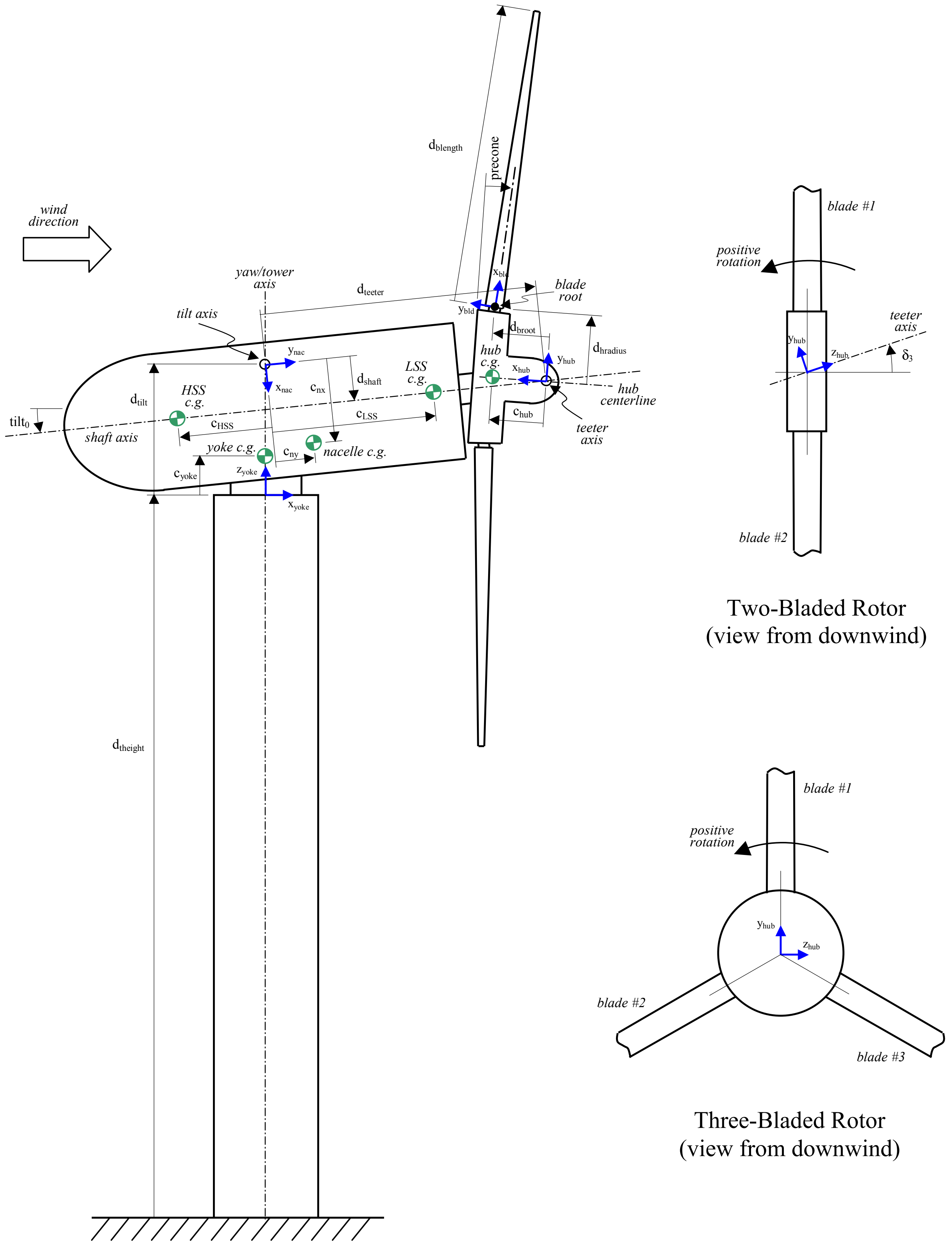
| | | |
|---|---|---|
| `mtop` | Lumped mass at the tower top | [kg] |

### Blade properties

The blade distributed properties are also defined by a table of data. Each row represents the properties at a given axial station. With reference to Figure 5, the table columns are:

| | | |
|---|---|---|
| $\bar{x}$ | Normalized station distance. 0.0 is blade root and 1.0 is blade tip. | - |
| m/L | Mass per-unit-length. | [kg/m] |
| $I_y$/L | Mass moment of inertia per-unit-length about the $y_{elast}$ axis. | [kg·m] |
| $I_z$/L | Mass moment of inertia per-unit-length about the $z_{elast}$ axis. | [kg·m] |
| $\theta_{elast}$ | Structural twist angle. | [deg] |
| $EI_y$ | Section bending rigidity about the $y_{bld}$ axis. | [N·m$^2$] |
| $EI_z$ | Section bending rigidity about the $z_{bld}$ axis. | [N·m$^2$] |
| chord | Chord length. | [m] |
| $\theta_{aero}$ | Aerodynamic twist angle. | [deg] |

At least two rows must be defined, for $\bar{x}$ = 0.0 and $\bar{x}$ = 1.0. Linear interpolation is used to find properties at intermediate stations.



Two-Bladed Rotor
(view from downwind)

Three-Bladed Rotor
(view from downwind)

**Figure 4: Geometry of input parameters**

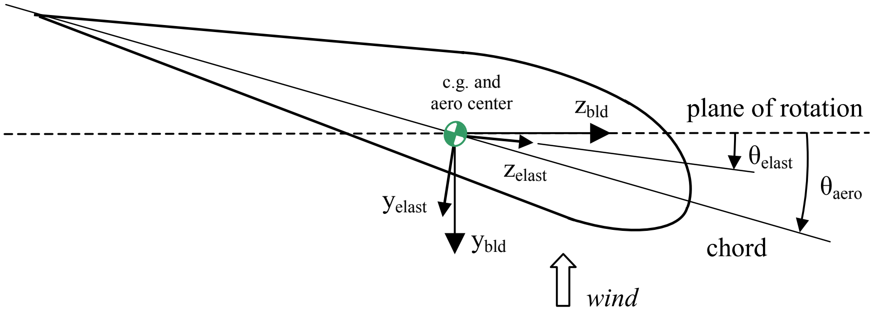


**Figure 5: Reference axes at a particular blade station. Shown with zero full-span pitch angle.**

The last input variable in the properties input file is `aero_elem_loc` , which determines the locations of the AeroDyn element centers along the blade length. The locations can be expressed as a fraction of the blade length in MATLAB vector form, e.g. `aero_elem_loc = [0.25, 0.6125, 0.875]`. Alternatively, a single integer value can be given, e.g. `aero_elem_loc = 20` , which is interpreted as the number of equal length elements desired. Improved aerodynamic results with low computation times are obtained when there is a higher density of AeroDyn elements nearer the blade tip than the root. The AeroDyn blade elements are illustrated for convenience in a plot when SymDynPP is executed. For the provided sample input with `aero_elem_loc = 20` the output plot will look like Figure 6(a). When a linear density function for the element centers is used, the output plot will look something like Figure 6(b).

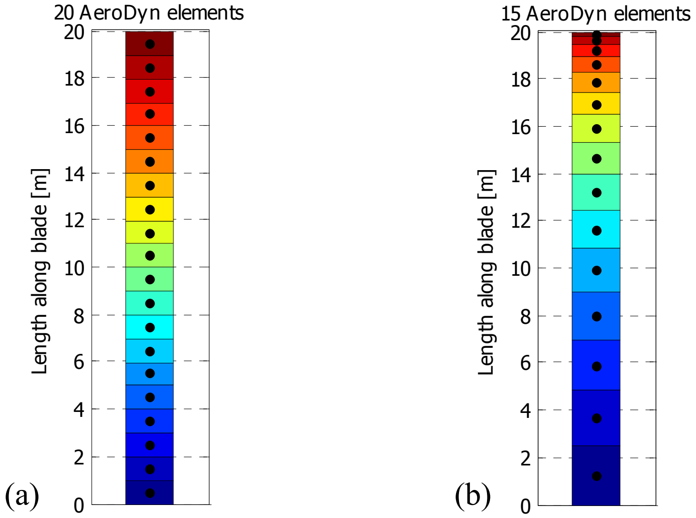


**Figure 6: Plots produced by the SymDyn preprocessor using different values for `aero_elem_loc` in *inputprops.m***

## 3.4 Simulation and Analysis Input File (*inputsim.m*)

The second input file shown in the flowchart of Figure 3 specifies parameters for all possible analysis cases, including simulation. This file must be named *inputsim.m*.

A sample *inputsim.m* file is given in Listing 3. Comment lines provide information about each parameter that is expected. Below are additional comments.

`active_dofs`
Lists the active DOFs in the structure, in MATLAB vector form. It must never be empty or contain a referenced DOF greater than $8+N_b$. When '6' is missing, this means the generator is running at constant (or zero) speed.

`aero_flag`
Determines whether aerodynamic loads are calculated. Without aerodynamics, many DOF combinations produce instability. 'calc_steady' will fail to run unless `aero_flag = 1` or joint damping is present.

`q0`
Vector of length $8+N_b$ determining the equilibrium positions for joint springs.

`q_init`
Vector of length $8+N_b$ determining both the initial displacements for active DOFs and prescribed displacements for inactive DOFs. E.g., if '4' is missing from active_dofs, fixed yaw position would be specified by `q_init(4)`.

`qdot_init`
Vector of length $8+N_b$ determining both the initial angular velocities for active DOFs and prescribed velocities for inactive DOFs. E.g. if '6' is missing from active_dofs, constant generator speed would be specified by `qdot_init(6)`.

Input variables for the calculation of a steady-state solution and for linearization will be described in the next two section.

## 3.5 Generating a Linearization Point (*calc_steady*)

A linear representation of an operating wind turbine is useful for modal analyses and in the design of linear control algorithms. The first step in linearizing the SymDyn model is choosing an operating condition to linearize about. It is in the neighborhood of this operating point that we expect our linear model to be valid.

There are three basic inputs to SymDyn that describe a particular operating condition:

1. Wind field
2. Blade pitch angles
3. Applied generator torque

We are assuming that generator torque is an independent input that is not prescribed by some control algorithm or generator dynamics.

An operating point is found by choosing time-invariant inputs and solving the nonlinear system in time until a steady-state solution is reached. This procedure is done automatically by the ‘calc_steady’ command described later. In general, the steady-state solution found will be periodic in time, with period equal to the time of one rotor revolution. We choose a wind field characterized by hub-height wind parameters and pitch angles that are equal for each blade. The operating point inputs must be defined in the *inputsim.m* file, as below.

`wdata_op`
: Vector of length 7 defining an AeroDyn hub-height wind field, see ref. [1], for the operating point.
`wdata_op(1)` = horizontal hub-height wind speed [m/s],
`wdata_op(2)` = wind direction with 0º aligned with 0º yaw angle [deg],
`wdata_op(3)` = vertical wind speed over the entire rotor disc [m/s],
`wdata_op(4)` = linear horizontal wind shear parameter,
`wdata_op(5)` = vertical shear power law exponent,
`wdata_op(6)` = linear vertical shear parameter,
`wdata_op(7)` = horizontal gust speed over the entire rotor disc [m/s].

`theta_op`
: Collective pitch angle for the operating point [rad].

`Tg_op`
: Applied generator torque for the operating point in the low-speed-shaft frame [N·m].

The operating point inputs determine the steady-state dynamics of the model, including the generator speed (which, in general, is periodic in time). However, the user would usually like to specify the mean generator speed in defining the linearization point. One way of doing this is to model the turbine as a constant speed machine. This is done by ensuring azimuth position is not an active DOF (i.e. removing ‘6’ from `active_dofs`) and specifying the desired constant speed in `qdot_init(6)`. `Tg_op` could be any value because it is ignored.

For a variable speed machine with a desired operating generator speed, we must compute a *trim* solution. We define trim to mean converting one of the prescribed inputs (either `theta_op` or `Tg_op`) into a control variable in order to satisfy the desired generator speed condition. The following parameters in *inputsim.m* define the trim solution.

`omega_des`
: The target mean generator speed (in the low-speed-shaft frame) for the trim solution [rad/s].

`trim_case`
: Determines how ‘calc_steady’ finds a periodic steady-state solution for a variable-speed model.
`trim_case = 1` means generator torque is adjusted (with constant pitch of `theta_op`) until the mean generator speed is equal to `omega_des`. `Tg_op` is used as an initial guess.
`trim_case = 2` means collective blade pitch is adjusted (with constant generator torque of `Tg_op`) until the mean generator speed is equal to `omega_des`. `theta_op` is used as an initial guess.

The calc_steady command, provided with the SymDyn code, will automatically compute the steady-state or trim solution. At the MATLAB command prompt, the command is entered as

```
>> calc_steady
```

Optional numerical arguments may also be given as follows:

```
>> calc_steady(debugflag, etol_0, maxiter, gains)
```

`debugflag` Flag specifying whether additional data and an update plot is output, for debugging purposes (0 = false, 1 = true, default: 0).

`etol0` Starting error tolerance for steady-state convergence testing (default: 1e-3).

`maxiter` Maximum number of inner simulation iterations before the analysis is stopped (default: 10).

`gains` Gain pair used in finding a trim solution (default: [3,6] when `trim_case = 1`, [2,6] when `trim_case = 2`).

Generally, the default arguments are sufficient. If calc_steady fails to find a solution for the variable speed case, then the user should first set the debugflag and examine the updated output plot. If it appears that the solution is stable but not converging then increase `maxiter`. If the solution appears unstable, then decrease `gains` and try again.

Once successful, calc_steady saves the operating point data to a file (e.g. *opsave_test.mdl*) for use by the linearization routine. The periodic steady-state solution is discretized in time based on the `nsteps` input parameter.

`nsteps` Number of steps to discretize the periodic steady-state solution and state matrices into. At least 200 steps is recommended.

## 3.6 Generating State-Space Matrices (*calc_ABCD*)

The linear SymDyn model has the form

$$\begin{cases} \dot{\underline{x}} = A(t)\ \underline{x} + B(t)\ \underline{u} + B_d(t)\ \underline{u}_d \\ \underline{y} = C(t)\ \underline{x} + D(t)\ \underline{u} + D_d(t)\ \underline{u}_d \end{cases}$$

where $\underline{x}$ is the vector of state variables defined by

$$\underline{x} = \begin{bmatrix} \Delta\underline{q} \\ \Delta\dot{\underline{q}} \end{bmatrix},$$

$\Delta\underline{q}$ are perturbations of the DOFs from the operating point,

$\Delta\dot{\underline{q}}$ are perturbations of the DOF rates from the operating point,

$\underline{u}$ is the vector of control inputs (also perturbations),

$\underline{u}_d$ is the vector of disturbance inputs (also perturbations),

$\underline{y}$ is the vector of measured outputs (also perturbations),

A, B, $B_d$, C, D, $D_d$ are the periodic state-space matrices.

The number of DOFs defining the linear model is determined by the `active_dofs` list when calc_steady was executed. The lengths of $\underline{u}$, $\underline{u}_d$, and $\underline{y}$ are determined by the following input parameters in *inputsim.m*.

`torque_ctrl_swtch`

Determines whether generator torque is a control input in $\underline{u}$.
`torque_ctrl_swtch = 0` means no,
`torque_ctrl_swtch = 1` means yes (one input).

`pitch_ctrl_swtch`

Determines the number of pitch control inputs in $\underline{u}$.
`pitch_ctrl_swtch = 0` means no pitch inputs,
`pitch_ctrl_swtch = 1` means collective pitch (one additional input),
`pitch_ctrl_swtch = 2` means individual pitch ($N_b$ additional inputs).

`wdata_dist`

List of hub-height wind parameters to be treated as disturbance inputs in $\underline{u}_d$. Contains elements from the list `[1,2,3,4,5,6,7]`.

E.g. `wdata_dist = [1,5]` means perturbations of horizontal wind speed and vertical shear power law exponent are the disturbance variables.

`load_meas`

List of internal load measurement locations for use in constructing $\underline{y}$. Contains elements from the list `[1,2,3,4,5,6,7,8,...,8+` $N_b$`]`. Generally, these are SymDyn joint reference numbers, with the following exceptions:

`[1,2,3]` represent a location on the tower at a distance of `twr_sg_height` up from the base,

`[9,...,8+`$N_b$`]` represent the blade roots.

`load_meas_compt`

Boolean matrix to determine the load components for each measurement location, for use in constructing $\underline{y}$. Contains 6 columns and as many rows as elements in `load_meas`. The 6 columns represent [$F_x$, $F_y$, $F_z$, $M_x$, $M_y$, $M_z$], the Cartesian components of internal force and moment vectors. The reference frames are consistent with those illustrated in Figure 13 (Appendix A). A '1' entry means that the load component is measured, while a '0' entry means that it is not.

E.g. `load_meas = [1,9];`
```
    load_meas_compt = [0,1,0,0,0,1; 0,0,0,0,0,1];
```

`twr_sg_height`

Height above the tower base (ground) of the strain gauges that measure tower loads, i.e. for locations [1,2,3] of `load_meas`. [m]

In the control input vector, $\underline{u}$, the generator torque perturbation appears first followed by any blade pitch perturbations. For example,

if `torque_ctrl_swtch = 1` and `pitch_ctrl_swtch = 2` for a 3-bladed model, then

$$\underline{u} = \begin{bmatrix} \Delta T_g \\ \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta\theta_3 \end{bmatrix}.$$

The elements of $\underline{y}$ are constructed by assembling all nonzero columns of `load_meas_compt` for every entry in `load_meas`. For example,

if `load_meas = [1,9]` and `load_meas_compt = [0,1,0,0,0,1; 0,0,0,0,0,1]` then

$$\underline{y} = \begin{bmatrix} \Delta F_y\,(\text{twr fore - aft}) \\ \Delta M_z\,(\text{twr fore - aft}) \\ \Delta M_z\,(\text{bld \#1 root}) \end{bmatrix}.$$

As it is currently formed, $\underline{y}$ contains only measured loads. The user is able to measure any of the state variables in $\underline{x}$ by simply appending Boolean rows to the `C` matrix. This operation can be performed in the control design process.

The calc_ABCD command will automatically generate the periodic state-space matrices using the following MATLAB command.

```
>> calc_ABCD
```

An optional argument may also be given:

```
>> calc_ABCD(debugflag)
```

`debugflag` | Flag specifying whether plots will appear showing the results from an intermediate smoothing operation. Used for debugging purposes (0 = false, 1 = true, default: 0).

Once successful, calc_ABCD saves the state matrices to a file (e.g. *ABCD_test.mdl*) for use in modal analyses or control designs. The periodic matrices are discretized in time based on the `nsteps` input parameter, producing three-dimensional MATLAB arrays, e.g. the dimension of `A` will be (number of states) ➨ (number of states) ➨ (`nsteps`).

## 3.7 Simulation Output

One of the powerful tools available with a SymDyn model is the simulation of aeroelastic dynamics within MATLAB. Simulations of open- and closed-loop systems are performed in Simulink, MATLAB's graphical simulation environment, using SymDyn subsystem blocks and a combination of blocks from Simulink's library. The user can design his/her own control system inside Simulink and test it on the SymDyn nonlinear model.

The Simulink model in the *openloop_test.mdl* file is illustrated in Figure 7. This particular system can be used to generate the open-loop response of a SymDyn model. At the top of the figure is the parent system, which appears on first opening the file in Simulink. The important child subsystems are shown in the figure, bordered by dashed lines. Blocks that are shaded yellow represent data streams that are output to the MATLAB workspace. These signals form arrays, with each row generated at each major time step. An explanation of each output is given below ('ndof' is the number of active DOFs, 'ntsteps' is the number of simulation output time steps).

q_out
Angular displacements of the active DOFs.
Dimension: ntsteps➨ndof.

qdot_out
Angular velocities of the active DOFs.
Dimension: ntsteps➨ndof.

qdotdot_out
Angular accelerations of the active DOFs.
Dimension: ntsteps➨ndof.

theta_out
Blade pitch angles.
Dimension: ntsteps➨$N_b$.

Tg_out
Generator torque.
Dimension: ntsteps➨1.

aloads_out
Aerodynamic loads.
Dimension: ntsteps➨(12➨$N_b$).

With reference to Figure 13 and Figure 14 (Appendix A), aero loads on the flapping (outboard) section of blade #1 are applied at the flap hinge, in the {9} frame. Aero loads on the inboard section of blade #1 are applied at the root, in the {root1} frame.

The components of 'aloads_out' are ordered as follows.

| Column | Blade # | Component |
|---|---|---|
| 1 | 1 | $F_{b1x}$ |
| 2 | | $F_{b1y}$ |
| 3 | | $F_{b1z}$ |
| 4 | | $M_{b1x}$ |
| 5 | | $M_{b1y}$ |
| 6 | | $M_{b1z}$ |
| 7 | | $F_{h1x}$ |
| 8 | | $F_{h1y}$ |
| 9 | | $F_{h1z}$ |
| 10 | | $M_{h1x}$ |
| 11 | | $M_{h1y}$ |
| 12 | | $M_{h1z}$ |
| 13 | 2 | $F_{b2x}$ |
| 14 | | $F_{b2y}$ |
| 15 | | $F_{b2z}$ |
| 16 | | $M_{b2x}$ |
| 17 | | $M_{b2y}$ |
| 18 | | $M_{b2z}$ |
| 19 | | $F_{h2x}$ |
| 20 | | $F_{h2y}$ |

| | | |
|---|---|---|
| 21 | | $F_{h2z}$ |
| 22 | | $M_{h2x}$ |
| 23 | | $M_{h2y}$ |
| 24 | | $M_{h2z}$ |
| 25 | 3 | $F_{b3x}$ |
| 26 | | $F_{b3y}$ |
| 27 | | $F_{b3z}$ |
| 28 | | $M_{b3x}$ |
| 29 | | $M_{b3y}$ |
| 30 | | $M_{b3z}$ |
| 31 | | $F_{h3x}$ |
| 32 | | $F_{h3y}$ |
| 33 | | $F_{h3z}$ |
| 34 | | $M_{h3x}$ |
| 35 | | $M_{h3y}$ |
| 36 | | $M_{h3z}$ |
| ... | | ... |
| 12➨$N_b$-11 | $N_b$ | $F_{b'Nb'x}$ |
| 12➨$N_b$-10 | | $F_{b'Nb'y}$ |
| ... | | ... |
| 12➨$N_b$ | | $M_{h'Nb'z}$ |

loads_out

Internal structural loads.

Dimension: ntsteps➨(6➨(8+$N_b$)).

With reference to Figure 13 (Appendix A), internal loads are calculated at joints 4 through 8, relative to frames {4} through {8} respectively. Internal loads for the tower are calculated at a distance of `twr_sg_height` up from the tower base, referenced in the {1}, {2}, or {3} frames. Internal loads for each blade are calculated at the root locations (in the {root} frame). The tower and blade load measurement locations are consistent with the typical siting of tower and blade strain gauges.

The components of 'loads_out' are ordered as follows

| Column | Ref. frame/location | Component |
|---|---|---|
| 1 | Tower fore-aft | $F_x$ |
| 2 | | $F_y$ |
| 3 | | $F_z$ |
| 4 | | $M_x$ |
| 5 | | $M_y$ |
| 6 | | $M_z$ |
| 7 | Tower side-to-side | $F_x$ |
| 8 | | $F_y$ |
| 9 | | $F_z$ |
| 10 | | $M_x$ |
| 11 | | $M_y$ |
| 12 | | $M_z$ |
| 13 | Tower twist | $F_x$ |
| 14 | | $F_y$ |

| | | |
|---|---|---|
| 15 | | $F_z$ |
| 16 | | $M_x$ |
| 17 | | $M_y$ |
| 18 | | $M_z$ |
| 19 | Nacelle yaw | $F_x$ |
| 20 | | $F_y$ |
| 21 | | $F_z$ |
| 22 | | $M_x$ |
| 23 | | $M_y$ |
| 24 | | $M_z$ |
| 25 | Nacelle tilt | $F_x$ |
| 26 | | $F_y$ |
| 27 | | $F_z$ |
| 28 | | $M_x$ |
| 29 | | $M_y$ |
| 30 | | $M_z$ |
| 31 | Generator azimuth | $F_x$ |
| 32 | | $F_y$ |
| 33 | | $F_z$ |
| 34 | | $M_x$ |
| 35 | | $M_y$ |
| 36 | | $M_z$ |
| 37 | Shaft torsion | $F_x$ |
| 38 | | $F_y$ |
| 39 | | $F_z$ |
| 40 | | $M_x$ |
| 41 | | $M_y$ |
| 42 | | $M_z$ |
| 43 | Hub teeter | $F_x$ |
| 44 | | $F_y$ |
| 45 | | $F_z$ |
| 46 | | $M_x$ |
| 47 | | $M_y$ |
| 48 | | $M_z$ |
| 49 | Blade #1 root | $F_x$ |
| 50 | | $F_y$ |
| 51 | | $F_z$ |
| 52 | | $M_x$ |
| 53 | | $M_y$ |
| 54 | | $M_z$ |
| 55 | Blade #2 root | $F_x$ |
| 56 | | $F_y$ |
| 57 | | $F_z$ |
| 58 | | $M_x$ |
| 59 | | $M_y$ |
| 60 | | $M_z$ |
| 61 | Blade #3 root | $F_x$ |
| 62 | | $F_y$ |

| | | |
|---|---|---|
| 63 | | $F_z$ |
| 64 | | $M_x$ |
| 65 | | $M_y$ |
| 66 | | $M_z$ |
| ... | | ... |
| 6➨(8+$N_b$)-5 | Blade #$N_b$ root | $F_x$ |
| 6➨(8+$N_b$)-4 | | $F_y$ |
| 6➨(8+$N_b$)-3 | | $F_z$ |
| 6➨(8+$N_b$)-2 | | $M_x$ |
| 6➨(8+$N_b$)-1 | | $M_y$ |
| 6➨(8+$N_b$) | | $M_z$ |

In addition to the output variables above, Simulink automatically generates the ‘tout’ variable, listing the time at each integration step. Built-in MATLAB plot commands can then be used to generate figures for any of the output variables versus time. Alternatively, the user could execute SymDyn’s *plotsim.m* script to generate plots of the primary variables.

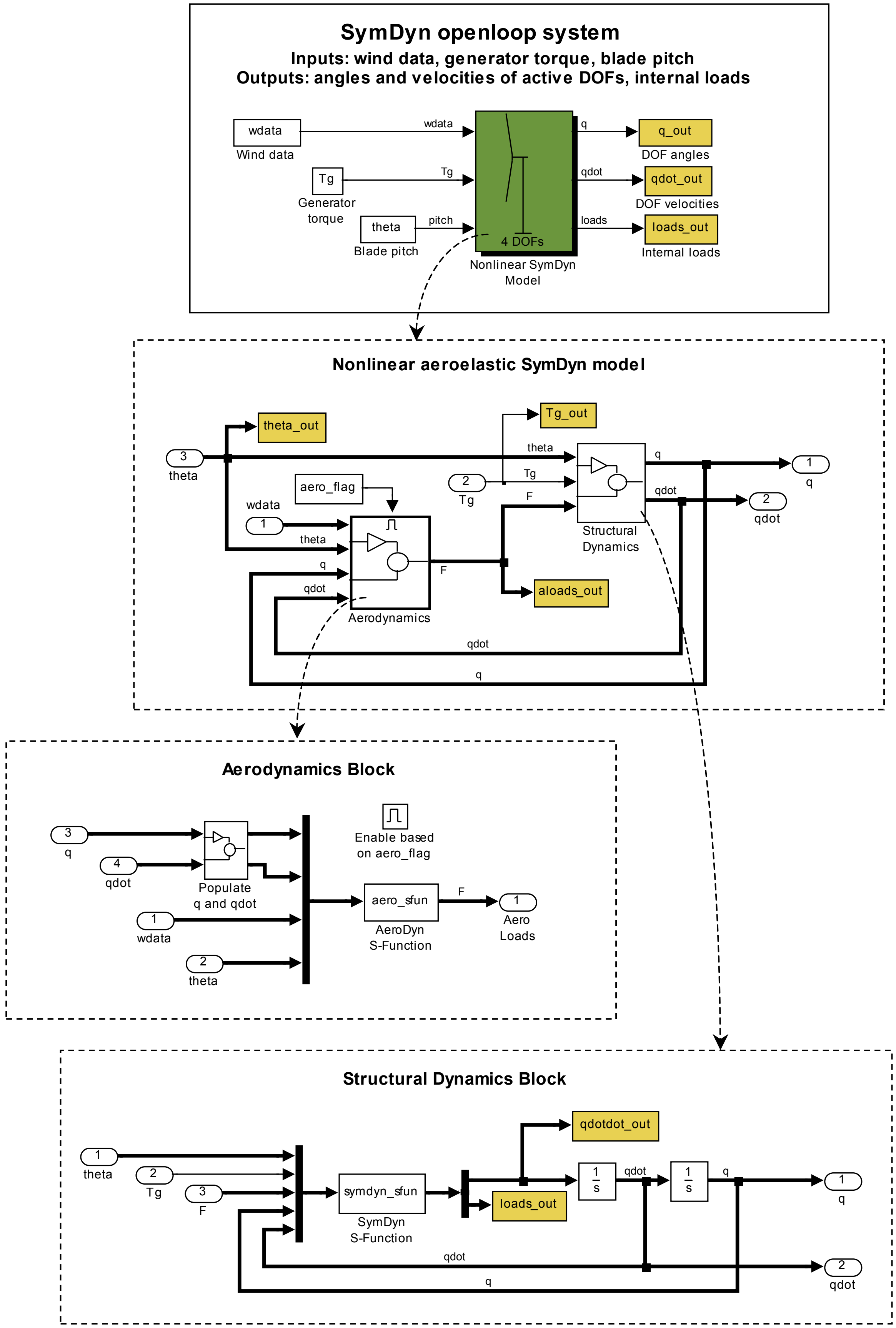


**Figure 7: Simulink model *openloop_test.mdl* and its major subsystems**

# 4. Example Study

The following instructions provide a quick-start to using SymDyn. A full analysis is conducted, including: the construction of a SymDyn model, linearization to form state-space matrices, stability analyses, and open- and closed-loop simulations. The user should refer to the design flow chart (Figure 3) as each step is completed.

('>> xxx' means 'xxx' is entered in the MATLAB command window)

## To generate SymDyn parameters from distributed wind turbine data:

1. Edit *inputprops.m* to specify all the wind turbine distributed properties. Leave the file unchanged (Listing 1) as an example.
2. >> SymDynPP
3. At the prompt, leave the response blank and press enter (i.e. use the default *inputprops.m* file).
4. Open '.\Intermediate_Files\*ADdata.ipt*' in a text editor. The file should match Listing 2. Copy and paste data to *aerodyn.ipt* if necessary. Add airfoil index numbers to form the last column of blade element data.

## To run an open-loop simulation:

1. Edit *inputsim.m* to specify the active DOFs, initial conditions, wind data and other simulation parameters. Leave the file unchanged (Listing 3) as an example.
2. Edit *aerodyn.ipt* to ensure all the AeroDyn options are specified as desired.
3. >> initialize
4. Review the screen output to verify the input parameters.
5. Open the Simulink file *openloop_test.mdl*, modify and add blocks as necessary (with caution), and run the simulation.
6. >> plotsim - to plot results. For the provided input files, the plots should match those in Figure 8.

## To calculate the steady-state solution:

1. Edit *inputsim.m* to specify the active DOFs, initial conditions, and other analysis parameters.
2. >> initialize
3. >> calc_steady - this may take a few minutes to complete. For the provided input files, the resulting plots should match those in Figure 9.
4. At the prompts, enter file names to write the operating point and initial condition data to (e.g. *opsave_test* and *ic_test*).

## To linearize and calculate the state-space matrices:

1. Edit *inputsim.m* to specify the active DOFs and other analysis parameters, consistent with the steady-state solution just found.
2. >> initialize

3. >> calc_ABCD
4. At the prompt, enter the desired operating point type. Entering '1' for 'periodic' (recommended) requires the steady-state solution file to be available. In this case, enter the file name at the prompt (e.g. *opsave_test*)
5. At the prompt, enter a file name to write the state-space matrices to (e.g. *ABCD_test*).

### To perform an open-loop stability analysis:

1. >> calc_modes
2. At the prompt, enter the file name to read the state-space matrices from (e.g. *ABCD_test*).
3. A conventional eigenanalysis on the mean of A(t) and a Floquet analysis on A(t) is performed. The conventional eigenanalysis results in eigenvalues (`eigval`) and corresponding eigenvectors (`eigvect`), and the Floquet analysis results in characteristic exponents (`CE`) and corresponding modeshapes (`MS`). These variables are available in the MATLAB workspace. For the provided input files, the output plots are shown in Figure 10.

### To design a simple full-state feedback controller and run a closed-loop simulation:

1. Edit *inputsim.m* to specify the active DOFs, initial conditions, wind data and other simulation parameters. The active DOFs must include those used in the calculation of the state-space matrices but may contain more.
2. Edit *aerodyn.ipt* to ensure all the AeroDyn options are specified as desired.
3. >> initialize
4. >> LQR_design
5. At the prompt, enter the file name for the saved state-space matrices (e.g. *ABCD_test*). Answer 'n' to 'Estimator design?'
7. Open the Simulink file *FSFB_test.mdl*, modify and add blocks if desired (with caution), and run the simulation.
8. >> plotsim - to plot results. For the provided input files, the plots should match those in Figure 11.

### To design a simple realizable controller and run a closed-loop simulation:

1. Edit *inputsim.m* to specify the active DOFs, initial conditions, wind data and other simulation parameters. The active DOFs must include those used in the calculation of the state-space matrices but may contain more.
2. Edit *aerodyn.ipt* to ensure all the AeroDyn options are specified as desired.
3. >> initialize
4. >> LQR_design
5. At the prompt, enter the file name for the saved state-space matrices (e.g. *ABCD_test*). Answer 'y' to 'Estimator design?'
6. Open the Simulink file *realizable_test.mdl*, modify and add blocks if desired (with caution), and run the simulation.
9. >> plotsim - to plot results. For the provided input files, the plots should match those in Figure 12.

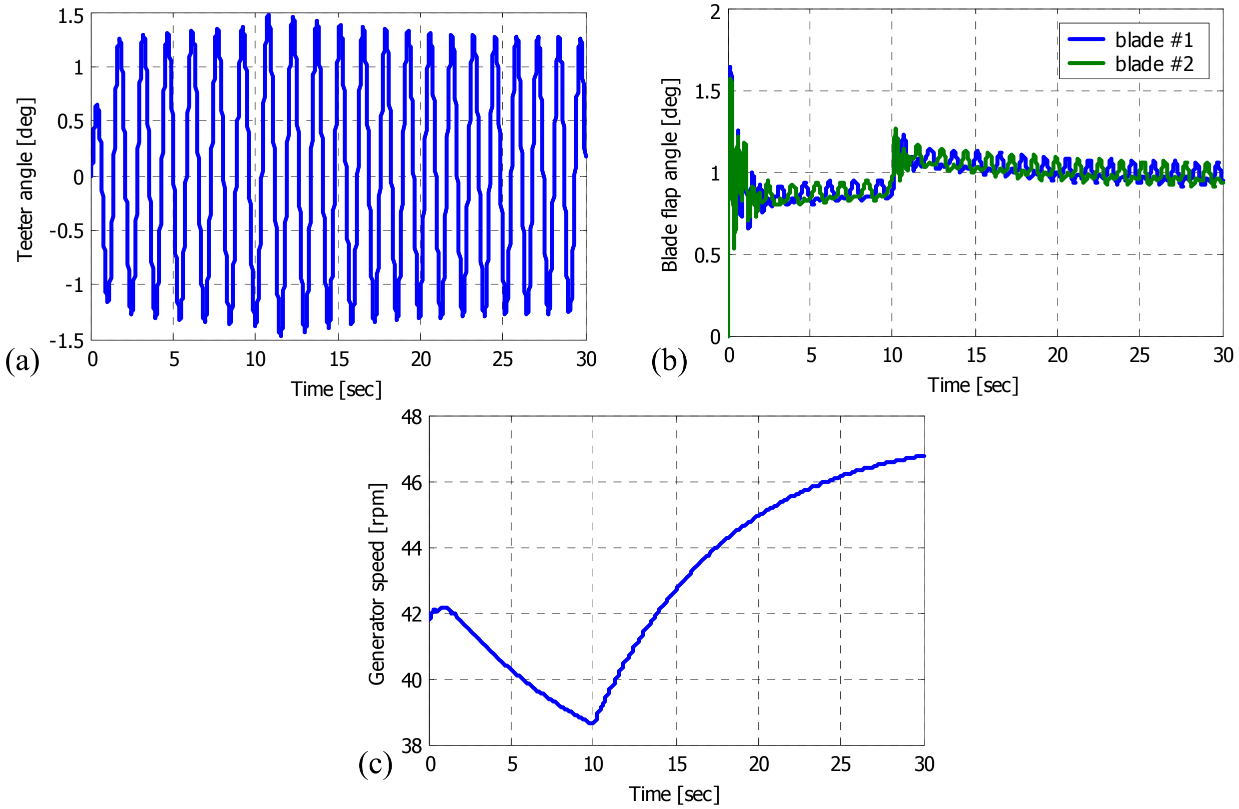


**Figure 8: Sample open-loop simulation plots**

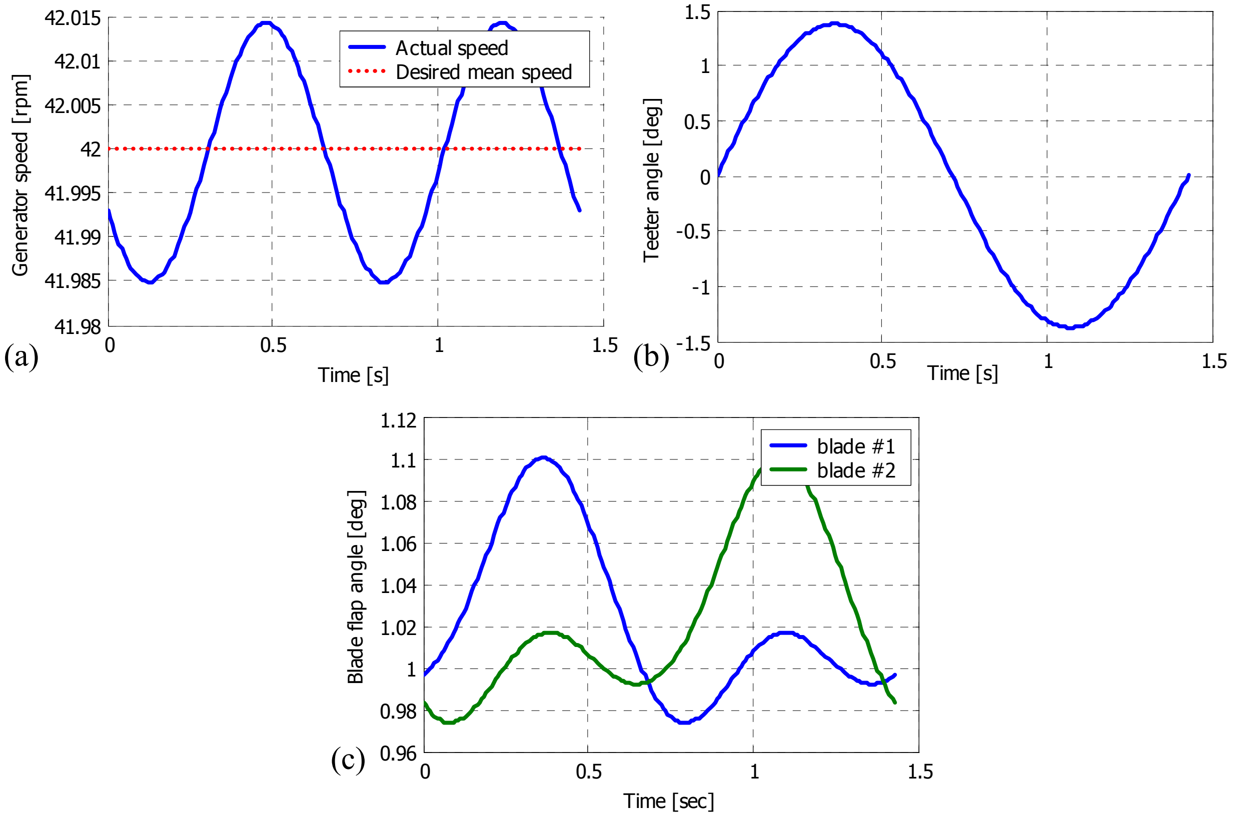


**Figure 9: Sample steady-state solution plots**

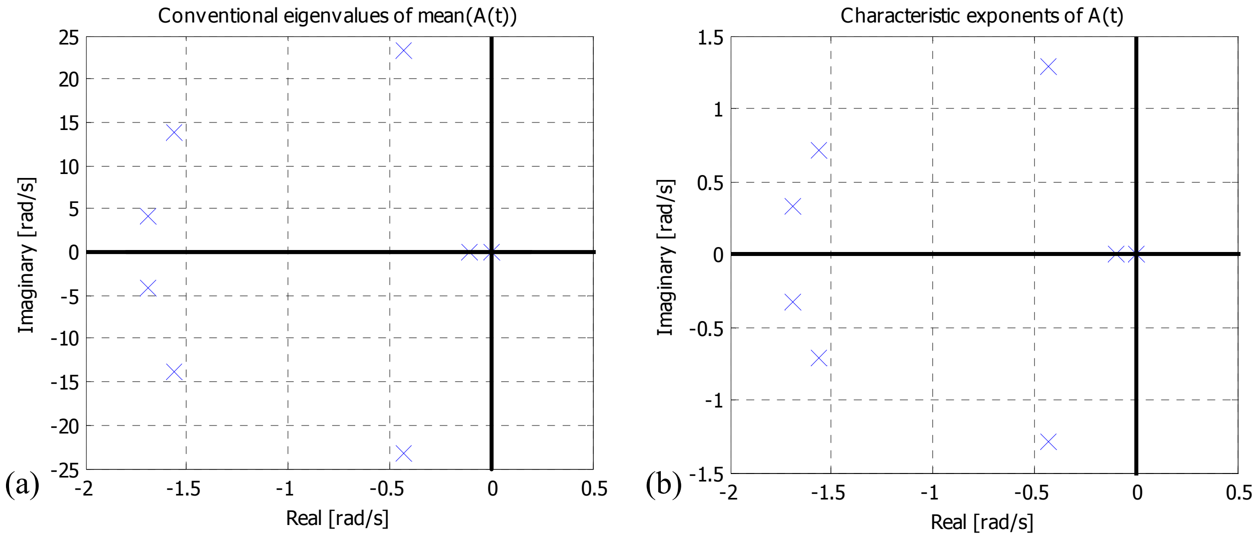


**Figure 10: Sample modal analysis plots**

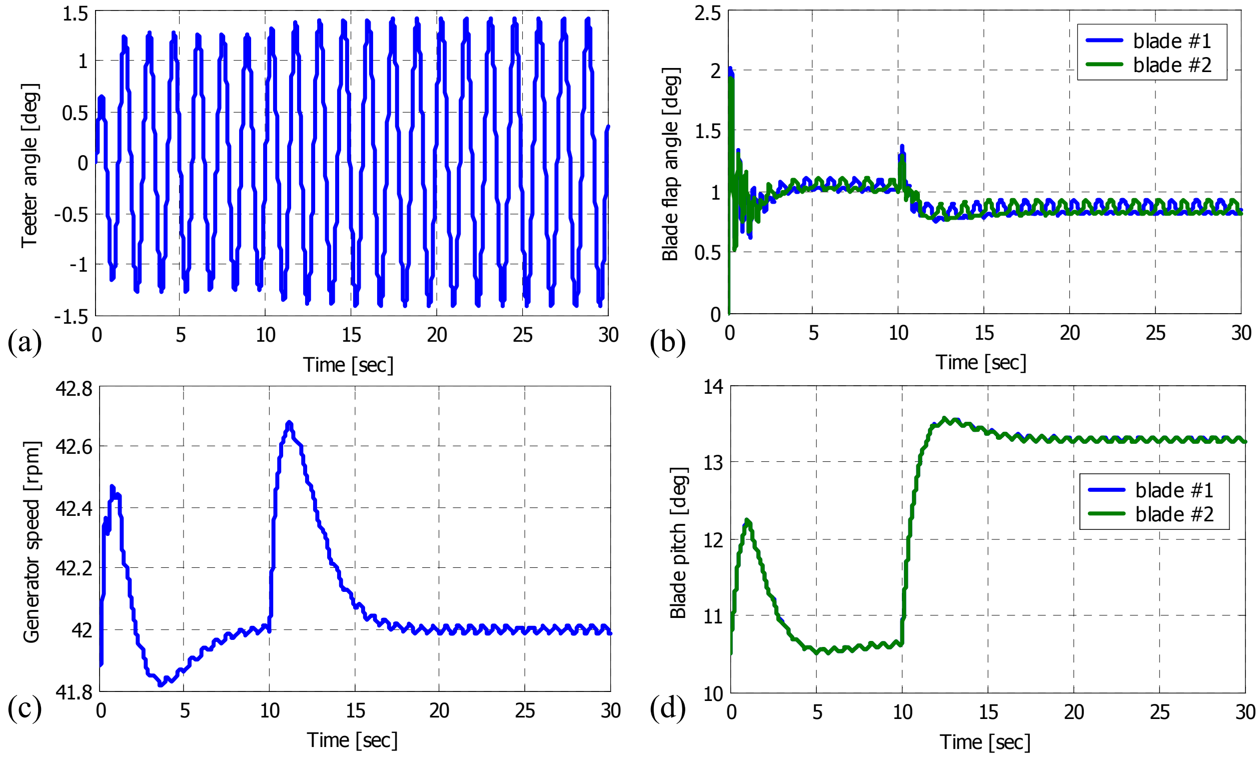


**Figure 11: Sample FSFB control plots**

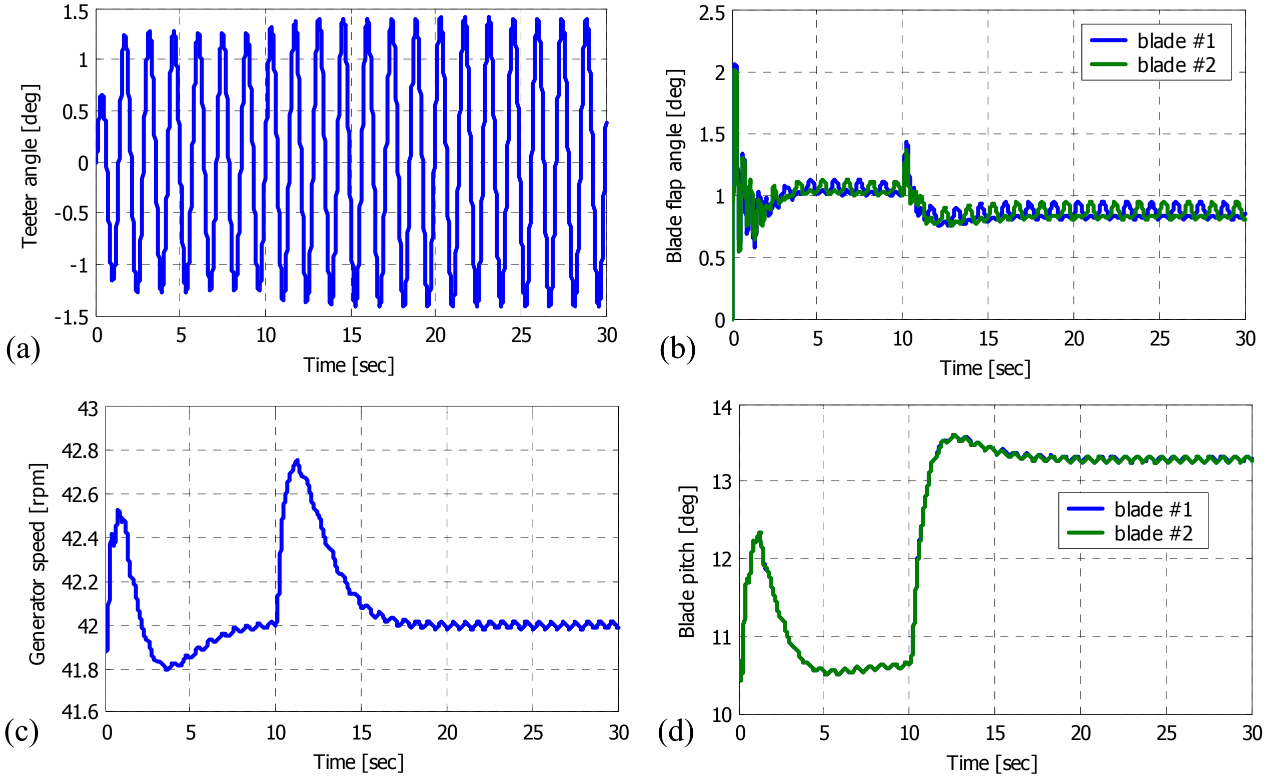


**Figure 12: Sample realizable control plots**

# 5. References

[1] Laino, D.J. and Hansen, A.C., 2002, "User's Guide to the Wind Turbine Aerodynamics Computer Software AeroDyn v12.50," *http://wind.nrel.gov/designcodes/aerodyn/aerodyn.pdf*, last accessed July 10, 2003.

[2] Stol, K., 2001, "Dynamics Modeling and Periodic Control of Horizontal-Axis Wind Turbines," Ph.D. Thesis, University of Colorado, Boulder, Colorado.

[3] Craig, J.J., 1955, *Introduction to Robotics*, Addison-Wesley Publishing Company, New York, NY.

# Appendix A: SymDyn Parameters

This section describes the SymDyn model that is constructed by the preprocessor algorithms. Under normal circumstances, the user is not required to understand these details.

For each rigid body there is an associated revolute joint that precedes it and a body-fixed reference system attached to it. The location and orientation of these coordinate systems follows the Denavit-Hartenberg convention. This convention is used in the robotics field to standardize the kinematics of open linkage systems [3]. The reference systems used in SymDyn are shown in Figure 13 with labels that correspond to the DOFs (e.g. "{5} Nacelle tilt"). The figure illustrates a 2-bladed turbine. Rotors with 3 or more blades have blade numbers consistent with the input properties described in Section 3.2 (see Figure 4). Revolute joints are located at each reference frame (with the exception of the inertial frame {0} and blade root frames {root1, root2, etc.}) with positive rotation about the positive z-axis of each.

SymDynPP generates the MATLAB-readable binary file *SDprops.mat* that contains all the parameters that constitute a SymDyn model. Included are joint location properties ($d_{t1}$, $d_{t2}$, $d_{n1}$, $d_{n2}$, $d_{h1}$, $d_{h2}$, $d_{broot}$, $d_{hradius}$) that are shown in Figure 13 and center-of-mass distances ($c_4$, $c_{5x}$, $c_{5y}$, $c_8$, $c_{9i}$, $c_9$), shown in Figure 14. Many of SymDyn's rigid bodies are either massless or have only mass moments-of-inertia defined, as listed in Table 1. All mass moments-of-inertia are defined about the center-of-mass, referenced in the part coordinate system. Each blade has identical mass properties.

| Body/frame ref. number | Associated DOF | Mass | Principle moments of inertia $[I_{xx}, I_{yy}, I_{zz}]$ |
|---|---|---|---|
| 1 | Tower fore-aft | 0 | [0, 0, 0] |
| 2 | Tower side-to-side | 0 | [0, 0, 0] |
| 3 | Tower twist | 0 | $[I_{30}, I_{30}, I_{3z}]$ |
| 4 | Nacalle yaw | $m_4$ | $[I_{4x}, I_{4y}, I_{4z}]$ |
| 5 | Nacelle tilt | $m_5$ | $[I_{5x}, I_{5y}, I_{5z}]$ |
| 6 | Shaft rotation | 0 | $[0, 0, I_{6z}]$ |
| 7 | Shaft compliance | 0 | $[0, 0, I_{7z}]$ |
| 8 | Hub teeter | $m_8$ | $[I_{8x}, I_{8y}, I_{8z}]$ |
| root1 | - | $m_{9i}$ | $[0, I_{9i}, I_{9i}]$ |
| 9 | Blade #1 flap | $m_9$ | $[0, I_{90}, I_{90}]$ |
| ... | ... | ... | ... |
| root2 | - | $m_{9i}$ | $[0, I_{9i}, I_{9i}]$ |
| 10 | Blade #$N_b$ flap | $m_9$ | $[0, I_{90}, I_{90}]$ |

**Table 1: SymDyn mass properties**

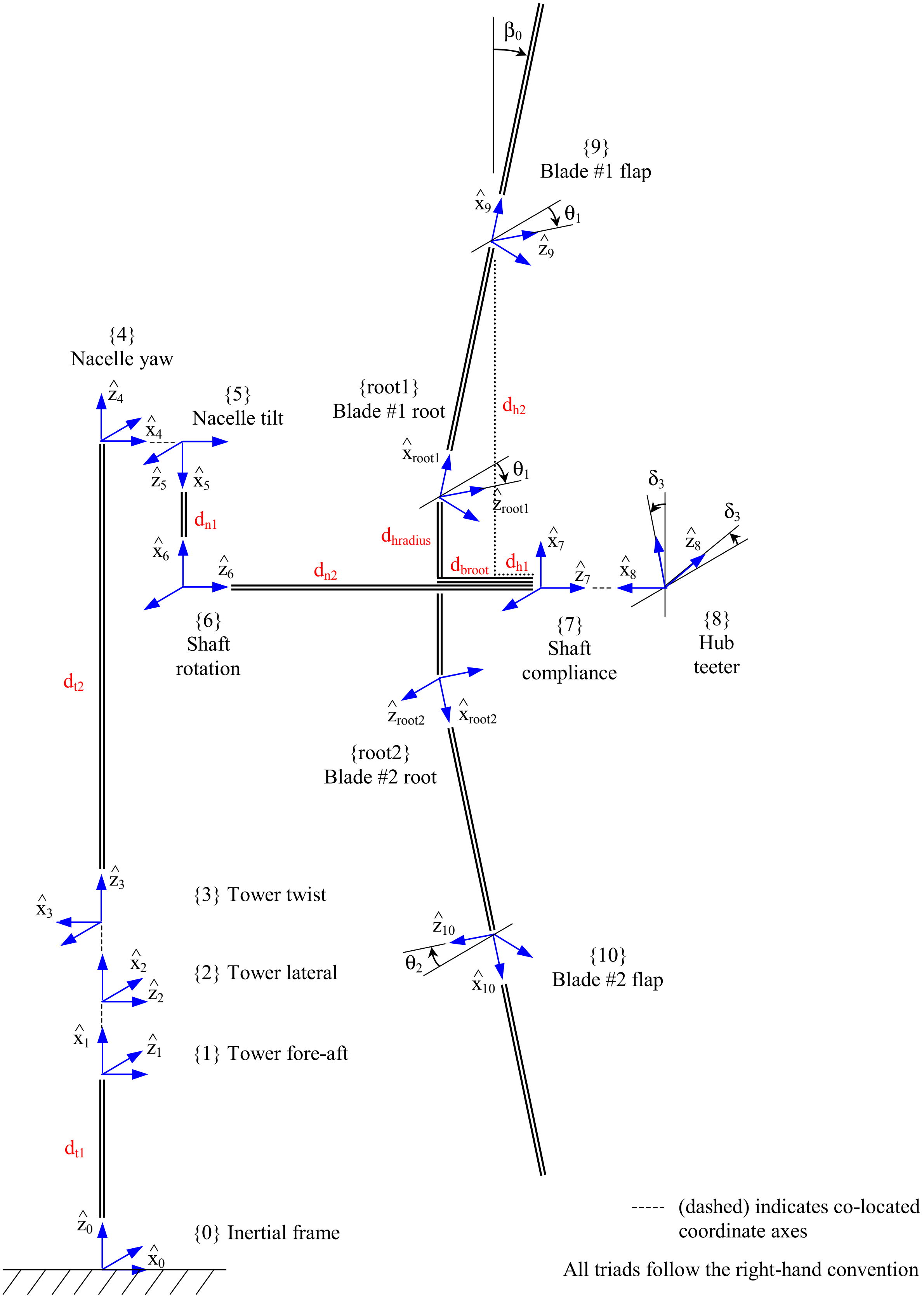


**Figure 13: SymDyn reference systems and joint locations. Shown for a 2-bladed turbine, with all angular degrees of freedom set to zero.**

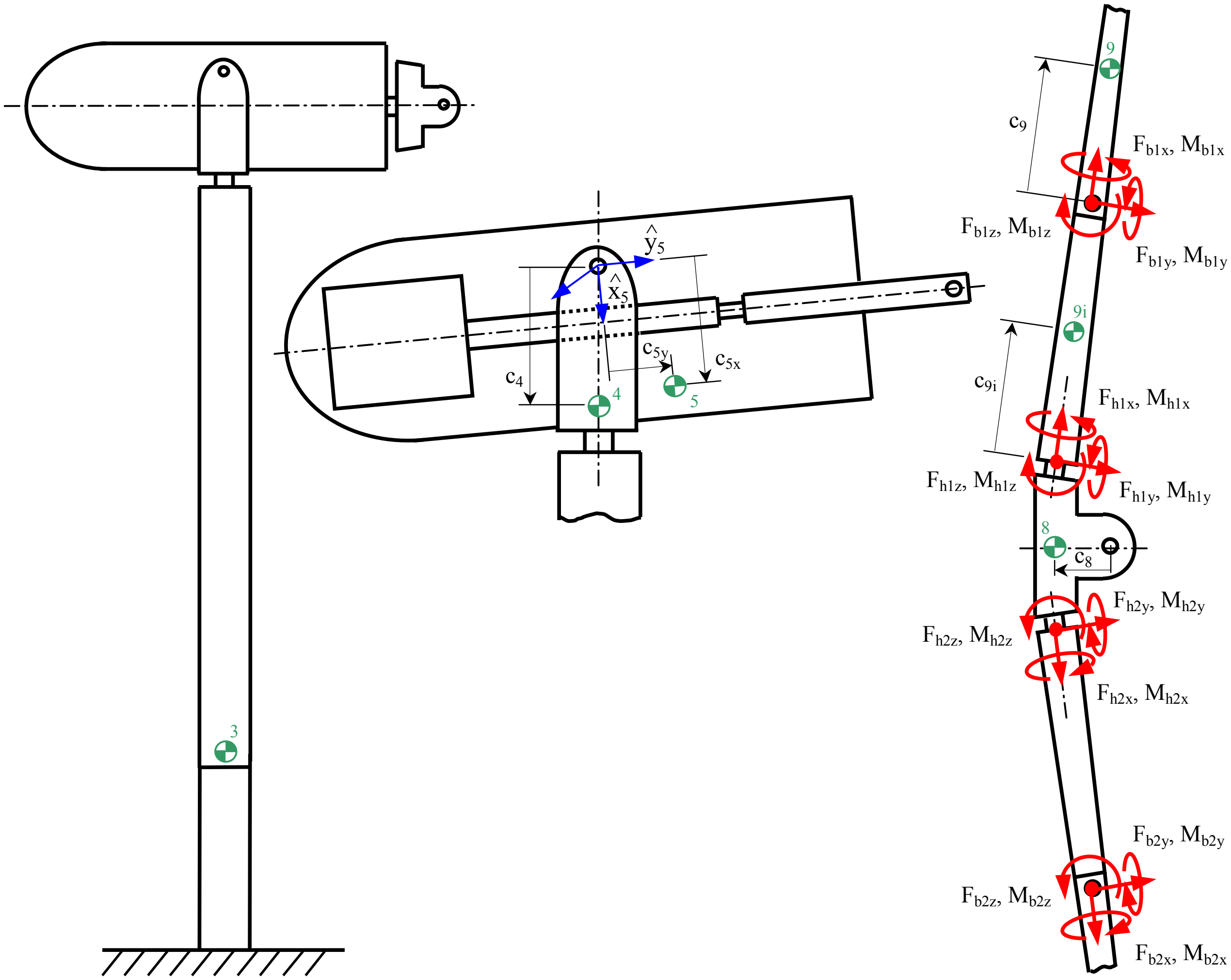


**Figure 14: SymDyn center-of-mass and aerodynamic load locations. Shown for a 2-bladed turbine.**

# Appendix B: Sample File Listings

```
% inputprops.m:  Contains the input turbine properties for derivation of SymDyn parameters
%                  via the SymDyn preprocessor (SymDynPP.m)
%   SymDyn v1.20 7/11/03
%   Assumes S.I. units
%

ftitle    = 'CART properties (6/02)';          % title for reference

% Geometry and other constants

Nb        = 2;           % number of blades
rigid_hub = 0;           % 0 = free teeter, 1 = locked teeter (for use in frequency matching)
gearratio = 43.165;      % gearbox gear ratio
precone   = 0;           % blade precone, pos. moves blade tips downwind [deg]
tilt0     = -3.77;       % nominal tilt, pos. moves downwind end of nacelle up [deg]
delta3    = 0;           % teeter axis angular offset (ignored for locked teeter or Nb>2) [deg]
omega0    = 42;          % nominal rotor speed, pos. clockwise when looking downwind [rpm]

dtheight  = 34.862;      % tower height [m]
dtilt     = 1.734;       % height from tower top to tilt axis, pos. up [m]
dshaft    = 0;           % dist. from tilt axis to shaft axis, normal to shaft, pos. down [m]
dteeter   = -3.867;      % dist. from tilt axis to teeter axis, parallel to shaft, pos. downwind [m]
dhradius  = 1.381;       % dist. from teeter axis to blade root, normal to hub centerline [m]
dbroot    = 0;           % dist. from teeter axis to blade root, parallel to hub centerline [m]
dblength  = 19.9548;     % blade length from root to tip [m]

% Center of mass locations

cyoke     = 0;           % c.g. of nacelle yoke, measured up from tower top along yaw axis [m]
cnx       = 0;           % c.g. of nacelle, measured down from tilt axis [m]
cny       = -0.402;      % c.g. of nacelle, measured downwind from tilt axis [m]
cHSS      = 0;           % c.g. of HSS + generator from tilt axis along shaft, pos. upwind [m]
cLSS      = -3.867;      % c.g. of LSS from tilt axis along shaft, pos. downwind [m]
chub      = 0;           % c.g. of hub from teeter axis, measured upwind along hub center [m]

% Masses

myoke     = 0;           % mass of nacelle yoke [kg]
mnac      = 23228;       % mass of nacelle + nonrotating parts of generator and shaft bearings [kg]
mHSS      = 0;           % mass of HSS + rotating generator parts [kg]
mLSS      = 5885;        % mass of LSS [kg]
mhub      = 5852;        % mass of hub [kg]

% Moments of inertia (MOI's)

Iyokex    = 0;           % MOI of nacelle yoke in {yoke} frame [kg.m^2]
Iyokey    = 0;           % "
Iyokez    = 0;           % "
Inacx     = 3.659e4;     % MOI of nacelle and all nonrotating gen. parts in {nac} frame [kg.m^2]
Inacy     = 1.2e3;       % "
Inacz     = 3.659e4;     % "
IHSSlat   = 0;           % lateral MOI of HSS + generator [kg.m^2]
IHSSlong  = 34.4;        % longitudinal MOI of HSS + generator [kg.m^2]
ILSSlat   = 0;           % lateral MOI of LSS [kg.m^2]
ILSSlong  = 0;           % longitudinal MOI of LSS [kg.m^2]
Ihubx     = 1.5e4;       % MOI of hub in {hub} frame [kg.m^2]
Ihuby     = 0;           % "
Ihubz     = 1.5e4;       % "

% Joint and shaft stiffnesses

kyaw      = 0;           % yaw joint torsional stiffness [N.m/rad]
ktilt     = 0;           % tilt joint torsional stiffness [N.m/rad]
kteet     = 0;           % teeter torsional stiffness (ignored for rigid hub or Nb>2) [N.m/rad]
kLSS      = 2.690e7;     % LSS torsional stiffness (value <= 0 interpreted as rigid) [N.m/rad]
kHSS      = -1;          % HSS torsional stiffness (value <= 0 interpreted as rigid) [N.m/rad]
```

```
% Tower distributed properties
%  { x/height (m), mass-per-unit-length (kg/m), I/L (kg.m), GJ (N.m^2), EI (N.m^2) }
%
% must contain at least two rows, one for x/height = 0.0 and one for x/height = 1.0

tdata = [
0.000  1548   3444   3.06E+10       8.31E+10
0.066  1361   2311   2.05E+10       5.58E+10
0.197  1428   1277   1.13E+10       3.09E+10
0.262  1311   742    6.57E+09       1.80E+10
0.329  1311   742    6.57E+09       1.80E+10
0.430  1311   742    6.57E+09       1.80E+10
0.514  878    482    4.28E+09       1.17E+10
0.614  878    482    4.28E+09       1.17E+10
0.698  878    482    4.28E+09       1.17E+10
0.782  599    317    2.81E+09       7.65E+09
0.881  599    317    2.81E+09       7.65E+09
0.966  1311   742    6.57E+09       1.80E+10
1.000  1311   742    6.57E+09       1.80E+10
];
mtop = 1610;        % lumped mass at tower top (part of tower not nacelle, e.g. for yaw bearings)

% Blade distributed properties
%  { x/length (m), mass-per-unit-length (kg/m), Iy/L (kg.m), Iz/L (kg.m), ea_twist (deg),
%     EIy (N.m^2), EIz (N.m^2), chord (m), aero_twist (deg) }
%
% must contain at least two rows, one for x/length = 0.0 and one for x/length = 1.0

bdata = [
0.000  282.92  29.47  12.33  3.44   2.83E+08       1.65E+08       1.143  3.44
0.022  290.24  33.11  11.97  3.37   3.18E+08       1.61E+08       1.196  3.37
0.053  261.88  34.19  10.57  3.27   3.28E+08       1.42E+08       1.268  3.27
0.114  201.28  31.97  7.35   3.08   3.07E+08       9.87E+07       1.411  3.08
0.175  186.52  35.48  5.82   2.88   3.40E+08       7.84E+07       1.555  2.88
0.236  169.1   35.67  4.41   2.69   3.42E+08       5.92E+07       1.699  2.69
0.300  149.28  29.02  3.38   2.45   2.78E+08       4.54E+07       1.637  2.45
0.364  133.19  24.71  2.54   2.21   2.37E+08       3.41E+07       1.575  2.21
0.427  111.74  17.58  1.86   1.91   1.69E+08       2.50E+07       1.494  1.91
0.491  96.86   14.34  1.33   1.61   1.38E+08       1.79E+07       1.412  1.61
0.554  78.57   9.81   0.92   1.24   9.40E+07       1.23E+07       1.331  1.24
0.618  65.03   7.54   0.61   0.86   7.25E+07       8.19E+06       1.250  0.86
0.682  49.68   4.87   0.38   0.38   4.67E+07       5.14E+06       1.168  0.38
0.745  37.59   3.40   0.23   -0.11  3.26E+07       3.02E+06       1.087  -0.11
0.809  25.01   1.98   0.12   -0.77  1.90E+07       1.62E+06       1.006  -0.77
0.873  16.01   1.35   0.06   -1.43  1.30E+07       8.68E+05       0.925  -1.43
0.936  10.73   0.92   0.03   -2.37  8.85E+06       4.68E+05       0.843  -2.37
1.000  6.02    0.71   0.02   -3.31  6.80E+06       2.09E+05       0.762  -3.31
];
aero_elem_loc = 20;   % list of AeroDyn element locations from the blade root as a fraction of
                      % blade length _OR_ an integer for the number of equilength elements per blade
```

**Listing 1: Sample properties input file (*inputprops.m*)**

```
==================== partial AeroDyn input data from SymDyn v1.20 =====================
=== copy the text below, paste to "aerodyn.ipt" and add extra columns for element data ===

AeroDyn (v12.52) input with CART properties (6/02)
SI     Units for input and output [SI or ENGlish]
 ...
36.850 Wind reference (hub) height.
 ...
20     Number of blade elements per blade
RELM       Twist      DR         Chord     File ID     Elem Data
0.4989     3.3603     0.9977     1.2030
1.4966     3.2015     0.9977     1.3196
2.4943     3.0439     0.9977     1.4370
3.4921     2.8800     0.9977     1.5550
4.4898     2.7243     0.9977     1.6730
5.4876     2.5438     0.9977     1.6612
6.4853     2.3562     0.9977     1.6128
7.4831     2.1576     0.9977     1.5609
8.4808     1.9195     0.9977     1.4966
9.4785     1.6850     0.9977     1.4325
10.4763    1.4103     0.9977     1.3683
11.4740    1.1153     0.9977     1.3044
12.4717    0.8075     0.9977     1.2410
13.4695    0.4325     0.9977     1.1770
14.4672    0.0456     0.9977     1.1127
15.4650    -0.4194    0.9977     1.0490
16.4627    -0.9350    0.9977     0.9858
17.4605    -1.4598    0.9977     0.9224
18.4582    -2.2059    0.9977     0.8573
19.4559    -2.9428    0.9977     0.7936
```

**Listing 2: Sample *ADdata.ipt* output file from the SymDyn preprocessor**

```
% inputsim.m:  Contains initialization data for analyses
%   SymDyn v1.20 7/11/03
%   Assumes S.I. units
%

%%
% General inputs
%%

active_dofs = [6,8,9,10]; % active degrees of freedom from the list [1,2,3,4,5,6,7,8,9,...8+Nb]
aero_flag = 1;            % aerodynamics flag (1 = aero on, 0 = aero off)
usewindfile_flag = 0;     % flag for use of AeroDyn wind file (1 = yes, 0 = no)
g = 9.81;                 % gravity acceleration [m/s^2]

%%
% Joint structural damping [N.m.s/rad]
%%

Cjoint(1) = 0;          % tower fore-aft damping
Cjoint(2) = 0;          % tower side-to-side damping
Cjoint(3) = 0;          % tower twist damping
Cjoint(4) = 0;          % yaw joint damping
Cjoint(5) = 0;          % tilt joint damping
Cjoint(6) = 0;          % generator shaft damping
Cjoint(7) = 0;          % shaft torsion damping
Cjoint(8) = 0;          % teeter joint damping
Cjoint(9) = 0;          % blade flap damping

%%
% Override SymDyn parameters if desired
%   Do not change Nb here (this must by done in inputprops.m and SymDynPP rerun)
%%

%tilt0 = 0;     % zero tilt (example)
%K4 = 1e6;      % nonzero yaw stiffness (example)
%K5 = 1e7;      % nonzero tilt stiffness (example)

%%
% Initial conditions and prescribed displacements and velocities
%%

% Equilibrium position for joints, when spring torque is zero [radians]
%  Fixed tilt and precone values are already included
q0 = zeros(1,8+Nb);         % zeros

% Initial conditions for joint angles [radians]
%  Fixed tilt and precone values are already included
q_init(1) = 0;              % tower fore-aft
q_init(2) = 0;              % tower lateral
q_init(3) = 0;              % tower twist
q_init(4) = 0;              % yaw
q_init(5) = 0;              % tilt
q_init(6) = 0;              % azimuth
q_init(7) = 0;              % shaft compliance
q_init(8) = 0;              % teeter
q_init(9) = 0;              % flap of blade #1
q_init(10) = 0;             % flap of blade #2
%q_init(11) = 0;            % flap of blade #3 - uncomment for 3-bladed rotor

% Initial conditions for joint velocities [radians/s]
qdot_init(1) = 0;           % tower fore-aft rate
qdot_init(2) = 0;           % tower lateral rate
qdot_init(3) = 0;           % tower twist rate
qdot_init(4) = 0;           % yaw rate
qdot_init(5) = 0;           % tilt rate
qdot_init(6) = 42*pi/30;    % generator speed
qdot_init(7) = 0;           % shaft compliance rate
qdot_init(8) = 0;           % teeter rate
qdot_init(9) = 0;           % flap rate of blade #1
qdot_init(10) = 0;          % flap rate of blade #2
%qdot_init(11) = 0;         % flap rate of blade #3 - uncomment for 3-bladed rotor
```

```
%%
% Inputs for calculation of steady-state operating point (using calc_steady.m)
%    and for linearization (using calc_ABCD.m)
% 'constant-speed' means azimuth position is not an active degree-of-freedom
%%

trim_case = 2;               % calc_steady.m case (1 = find gen torque, 2 = find coll. pitch)
                             %     - ignored for constant-speed case

% parameters for steady-state and linearization:
wdata_op = [18, 0, 0, 0, 0.2, 0, 0]; % operating pt hub-height wind data (delta in deg)
theta_op = 12*pi/180;                % operating pt collective blade pitch angle [rad]
Tg_op = 152129;                % operating pt gen. torque [Nm] - ignored for constant-speed case
omega_des = 42*pi/30;          % desired mean gen. speed [rad/s] - ignored for constant-speed case
nsteps = 200;                  % number of time steps to save operating point and state matrices over

% parameters for linearization only:
torque_ctrl_swtch = 0;      % Gen. torque control (0 = off, 1 = on)
pitch_ctrl_swtch = 2;       % Pitch control type (0 = no pitch, 1 = coll. pitch, 2 = individ. pitch)
wdata_dist = [1];     % Elements of AeroDyn HH wind data for treatment as disturbance, from [1,...,7]
load_meas = [9,10];         % List of load locations to define linear outputs, from [1,...,8+Nb]
load_meas_compt = [0,0,0,0,0,1];  % Boolean matrix for desired load components, one row for each
                   0,0,0,0,0,1]; %  location in load_meas, representing [Fx,Fy,Fz,Mx,My,Mz]
twr_sg_height = 9.3;         % Height of tower strain gauge from base for tower load measurements

%%
% Simulation inputs (custom user variables for Simulink models)
%     Typical variables are: wdata, theta, and Tg
%%

%wdata = [0.0, 18.0, 0, 0, 0, 0.2, 0, 0];   % custom wdata (steady wind)
wdata = [0.0, 18.0, 0, 0, 0, 0.2, 0, 0;      % custom wdata (step in wind speed)
         10.0, 18.0, 0, 0, 0, 0.2, 0, 0;
         10.01, 20.0, 0, 0, 0, 0.2, 0, 0;
         30.0, 20.0, 0, 0, 0, 0.2, 0, 0];
%wdata = [0.0, 16.0, 0, 0, 0, 0.2, 0, 0;     % custom wdata (ramp in wind speed)
%        10.0, 20.0, 0, 0, 0, 0.2, 0, 0];
theta = 12*pi/180*ones(Nb,1);                % custom pitch angles
Tg = 152129;                                 % custom generator torque
```

**Listing 3: Sample analysis input file (*inputsim.m*)**